Diseño heurístico de redes con requerimientos de conectividad

Héctor Cancela [1,2], Franco Robledo [1,2], Omar Viera[1]
[1] Universidad de la República
Address : Facultad de Ingeniería, J.Herrera y Reissig 565, Montevideo, Uruguay
Phone:+598-2-7114244 ext. 112, fax: +598-2-7110469
Email: [cancela,frobledo,viera]@fing.edu.uy
[2] IRISA/INRIA
Address : Campus de Beaulieu, Rennes 35042 CEDEX, France
Phone: +33-2-99847296, fax: +33-2-99847171
Email:[hcancela,frobledo]@irisa.fr

Resumen

Proponemos una heurística para el problema generalizado de Steiner (Generalized Network Steiner Problem), un problema de importancia en el diseño de redes de comunicaciones. Una instancia de este problema consiste en una red con costos asociados a las líneas y con requerimientos de arista-conectividad asociados a cada par (i,j) de nodos terminales. El objetivo es encontrar una red de mínimo costo utilizando las líneas disponibles y que satisfaga los requerimientos.

La heurística propuesta consiste en una búsqueda con componentes aleatorios, que emplea ideas provenientes de la metodología conocida como Sistemas de hormigas (Ant Systems).

Se presentan resultados experimentales sobre un conjunto de instancias del problema con diferentes características y requerimientos de conectividad, obteniendo en todos los casos resultados óptimos o muy cercanos al óptimo.

Palabras clave heurísticas; diseño topológico; arista-conectividad; problemas de Steiner.

1. Introducción

Cuando se diseña una red de telecomunicaciones, se pretende encontrar una topología de mínimo costo que debe satisfacer ciertos requerimientos de conexión entre pares de nodos de la red. Generalmente, estos requerimientos son elegidos de forma de mejorar la supervivencia de la red, es decir, su capacidad de resistir a fallas en sus componentes. Típicamente se establecen de antemano requerimientos de conectividad entre pares de nodos, y luego se buscan topologías de costo mínimo que satisfagan estos requerimientos mediante caminos disjuntos (arista disjuntos o bien nodo disjuntos) entre pares de nodos. En el caso más general, los requerimientos de conectividad pueden ser fijados independientemente para cada par de nodos (requerimientos de conectividad heterogeneos). Este problema es conocido como Problema Generalizado de Steiner

(Generalized Steiner Network Problem-GSP, o también Minimum Survivable Network Problem [5]).

Este problema puede ser formalizado de la siguiente manera. Dado un grafo simple no dirigido $G = (V, E)$, con V el conjunto de nodos y E el conjunto de aristas; y dados una matriz de costos reales positivos C asociados a las aristas, un subconjunto de nodos terminales $T \subseteq V$, y una matriz $R = \{r_{ij}\}_{i,j \in T}$ cuyos elementos son enteros positivos que indican los requerimientos de conectividad entre todo par de nodos de T, se desea encontrar un subgrafo G_T de G de costo mínimo, tal que para todo par de nodos $i, j \in T$, $i \neq j$, existen al menos r_{ij} caminos de aristas disjuntas (nodo disjuntos) que comunican i y j en G_T. A los nodos de $V \backslash T$ se les denomina nodos de Steiner. Ha sido demostrado por Krarup [13] que el problema GSP es NP-Completo tanto para requerimientos de arista-conectividad así como de nodo-conectividad. Algunas referencias sobre el GSP y problemas relacionados pueden encontrarse en [1, 2, 7, 13, 14].

Este artículo presenta una heurística para el GSP con requerimientos de arista-conectividad. El algoritmo diseñado, que se presenta en la Sección 2, es aplicable a la clase más general de grafos; consiste esencialmente en una heurística de tipo goloso para construir soluciones factibles al problema. Para introducir variedad en la búsqueda, la heurística utiliza internamente un procedimiento aleatorio de determinación de caminos que está inspirado en el método Ant Systems. Los detalles de este método se presentan en la Sección 3.

Para validar la heurística, realizamos una serie de experimentos en base a instancias test de distintas características. Estas instancias se presentan en la Sección 4; los resultados experimentales y las conclusiones del trabajo se incluyen en la Sección 5.

2. Heurística propuesta para el GSP

En esta sección presentamos una heurística para encontrar soluciones factibles y de bajo costo al GSP. Previamente introducimos las siguientes notaciones:

- Dado un grafo $G = (V, E)$, un subconjunto de nodos $N \subseteq V$, y un subconjunto de aristas $A \subseteq E$ con extremos en N, denotamos $G_s(N, A)$ al subgrafo de G inducido por N y A.
- Dado un grafo $G = (V, E)$ bajo las condiciones del GSP, denotamos como Γ_{GSP} al espacio de soluciones factibles asociado a dicha instancia. Dicho espacio esta dado por los subgrafos $G_s(N, A)$ tales que $T \subseteq N \subseteq V$, $A \subseteq E$, y $\forall i, j \in T$ existen al menos r_{ij} caminos arista-disjuntos entre i y j en $G_s(N, A)$.

El algoritmo propuesto, al cual denominamos A_GSP, recibe como entradas un grafo simple no dirigido $G = (V, E)$, el conjunto de nodos terminales $T \subseteq V$, la matriz C de costos asociados a las aristas, y la matriz R de requerimientos de conexión entre nodos terminales. En base a estos parámetros, el algoritmo busca una solución factible de bajo costo dentro del espacio Γ_{GSP}. La Figura 1 muestra el pseudocódigo del A_GSP; a continuación describimos a grandes rasgos su funcionamiento (una descripción y pseudocódigos más detallados pueden encontrarse en [8, 9]).

Iterativamente, mientras no se cumpla un cierto número máximo de iteraciones y no se haya encontrado una cierta cantidad de soluciones factibles, el algoritmo A_GSP realiza los siguientes pasos. Se inicializa una serie de parámetros, entre ellos la solución actual con $N = T$, $A = \emptyset$ (se toma $G_s(T, \emptyset)$ como grafo inicial) y $cost\_sol = \infty$. La matriz auxiliar $M = \{M_{ij}\}_{i,j \in T}$ almacena los requerimientos de conexión que faltan por satisfacer en la solución factible en construcción $G_s(N, A)$. La matriz auxiliar booleana $B = \{B_{ij}\}_{i,j \in T}$ indica que pares de nodos terminales de

T están comunicados en $G_s(N, A)$. El procedimiento Initialize_Requirements genera los valores iniciales para estas matrices: $M \leftarrow R$ y $B_{ij} \leftarrow$ FALSE, $\forall i, j \in T$.

Luego, $\forall i, j \in T$ se trata de encontrar la cantidad r_{ij} de caminos de aristas disjuntas que comuniquen ambos nodos terminales, construyéndose de esta manera una solución factible para la instancia del GSP. La determinación de cada uno de estos caminos arista-disjuntos se hace mediante un algoritmo que denominamos Search_Path, el cual calcula iterativamente un conjunto de caminos entre un par de nodos terminales sobre un el subgrafo de G que contiene a las aristas aún no empleadas en caminos entre estos nodos, y elige uno de ellos de acuerdo a un criterio de selección que tiene en cuenta tanto el costo del camino como la relación entre el costo y la utilidad del mismo. Este algoritmo se explica en más detalle en la Sección 3.

Una vez seleccionado un camino $\bar{w}$ por el algoritmo Search_Path, se actualiza la solución actual mediante las asignaciones $N = N \cup \text{NODES}(\bar{w})$, $A = A \cup \text{EDGES}(\bar{w})$ y $cost\_sol = cost\_sol + \text{COST}(\bar{w})$. Cada vez que se agrega un nuevo camino a la solución actual (en construcción), se controla si el costo de ésta es menor que el de la mejor solución factible encontrada hasta el momento. En caso afirmativo, se actualizan las matrices B y M mencionadas anteriormente, a través del procedimiento Update_Matrix. Este procedimiento tiene en cuenta, además de los terminales i y j, todos los otros nodos terminales, y actualiza los valores de M y de B para reflejar la información de todas las nuevas conexiones inducidas por p.

En caso que la solución generada no sea de menor costo que la mejor encontrada anteriormente, el algoritmo A_GSP recomienza el proceso e intenta construir una nueva solución factible desde el comienzo. De igual manera, si para algún par de nodos terminales $i, j \in T$ el algoritmo A_GSP no puede determinar mediante las aplicaciones del algoritmo Search_Path la cantidad r_{ij} de caminos de aristas disjuntas requerida, entonces el algoritmo A_GSP reinicia el ciclo e intenta construir una nueva solución factible a partir del grafo vacio. Si $\forall i, j \in T$ se logran encontrar r_{ij} caminos de aristas disjuntas, construyéndose así una solución factible $G_s(N, A) \in \Gamma_{GSP}$ con costo menor que el de la mejor solución encontrada hasta el momento, entonces el algoritmo A_GSP actualiza la mejor solución.

El algoritmo A_GSP retorna la mejor solución factible encontrada luego de un cierto número de iteraciones, así como el número de soluciones obtenidas (0 si la búsqueda no fue exitosa).

La complejidad del A_GSP es $O\left(r_{max} \times |V| \times \text{máx}\left\{|T|^2, \text{MAX\_ANTS}\right\}\right)$ donde r_{max} es el mayor requerimiento de conectividad, $r_{max} = \text{máx}\left\{r_{ij}\right\}_{i,j \in T}$, y MAX_ANTS es un parámetro del procedimiento Search_Path que regula la diversidad de la búsqueda.

3. Procedimiento Search_Path

El funcionamiento de la heurística presentada en la sección anterior depende de la búsqueda de caminos realizada por el procedimiento Search_Path.

Presentamos una posible alternativa de implementación para este procedimiento, basada en los sistemas de hormigas. Colorni, Dorigo y Maniezzo diseñaron una nueva metaheurística inspirados en la observación del proceso de optimización natural por el cual las hormigas de la especie *Linepithema Humile* son capaces de encontrar caminos cortos desde el nido hacia una fuente de alimentos [3]. Dicha metaheurística, denominada Ant System, puede ser adaptada para diferentes problemas de optimización combinatoria y está basada en la utilización de hormigas artificiales las cuales construyen en forma cooperativa soluciones factibles al problema. Los algoritmos de Ant System tienen como parámetros básicos: α (denominado sensibilidad del rastro), β (denominado sensibilidad de distancia), ρ (denominado coeficiente de evaporación de la feromona), y χ (cantidad de feromona dejada por una hormiga en cada movimiento). Un

```
Procedure A_GSP(V,E,T,C,R)

num_iter ← 0; /* Número de iteraciones */
num_sol ← 0; /* Número de soluciones factibles encontradas */
best_value ← ∞; /* Costo de la mejór solución factible */
while (num_iter ≤ MAX_ITER) and (num_sol < MAX_SOL) do
      /* Loop de búsqueda de soluciones factibles en el espacio Γ_GSP */
    N ← T; A ← ∅; cost_sol ← 0; /* Inicializo solución actual con G_s(T, ∅) */
    [M, B] ← Initialize_Requirements(R); /* Inicializo las matrices auxiliares M y B */
    OK ← TRUE; /* Variable indicatriz de busqueda exitosa */
    for each i, j ∈ T do
      while (M_ij > 0) and OK do /* Loop de búsqueda de caminos entre i y j */
        A_c ← E \ A; /* Considero las aristas que no pertenecen a G_s(N, A) */
        [p, numpaths] ← Search_Path(V, A_c, R, C, ant_param, i, j); /* Busco un
              nuevo camino arista disjunto entre i y j a partir del subgrafo G_s(V, A_c) */
        if numpaths > 0 then /* Si se encontró un nuevo camino actualizo G_s(N, A) */
          N ← N ∪ NODES(p); A ← A ∪ EDGES(p); cost_sol = cost_sol + COST(p);
          if (cost_sol < best_value) then
            [M, B] ← Update_Matrix(p, T, R, M, B); /* Actualizo M y B */
          else OK = FALSE; /* El costo de G_s(N, A) es superior a best_value */
          end_if;
        else
          OK = FALSE; /* No se encontró un nuevo camino entre i y j */
          exit_for_each;
        end_if;
      end_while;
    end_for_each;
    if OK then /* Actualizo la mejór solución factible hasta el momento con G_s(N, A) */
      N_sol ← N; A_sol ← A; best_value ← cost_sol;
      num_sol ← num_sol + 1;
      num_iter ← 0;
    else num_iter ← num_iter + 1;
    end_if;
end_while;
return N_sol, A_sol, num_sol;
end A_GSP;
```

Figura 1: Pseudocódigo de A_GSP.

descripción detallada de estas clases se encuentra en [3, 4, 12].

El pseudocódigo del algoritmo Search_Path diseñado en base a estas ideas se muestra en la Figura 2. El método recibe como entradas el subgrafo $G_s(V, A_c)$ (con $A_c = E \setminus A$), la matriz de requerimientos de conexión R, la matriz de costos C, los parámetros α, β, ρ y Q a utilizar, y dos nodos terminales $i, j \in T$. Retorna (si la búsqueda es exitosa) un camino p que comunica los terminales i y j en $G_s(V, A_c)$. El camino p encontrado será arista disjunto con los caminos ya encontrados que comunican i y j en $G_s(N, A)$.

Para buscar caminos, opera de la siguiente forma. Sea $G_s(V, E \setminus A)$ el subgrafo de G sobre el cual se desea determinar un nuevo camino entre dos nodos terminales i y j. Para ello se ubica un cierto número de hormigas artificiales en los nodos i y j respectivamente (la cantidad de hormigas a ubicar en cada nodo dependerá de r_{ij} y del grado del nodo en el subgrafo en cuestión), luego las hormigas se "mueven"sobre el subgrafo encontrándose una cierta cantidad de caminos que unen ambos nodos terminales. Cualquiera de los caminos encontrados será arista-disjunto con los caminos que ya comunican i con j en $G_s(N, A)$ (la solución en construcción). Sobre el conjunto de caminos que unen i con j encontrados por las hormigas, se selecciona aquél que satisfaga un Criterio de Selección de Camino, que está definido en la Subsección 3.8.

En las subsecciones siguientes incluimos detalles de varias subrutinas de este procedimiento.

3.1. Initialize_Prob_Trans

Recibe como entrada el subgrafo $G_s(V, A_c)$, y devuelve la matriz estocástica $MP^{(ini)}$ de dimensión $n \times n$, con las probabilidades iniciales de transición para las hormigas. La matriz es calculada de la siguiente forma:

$$MP_{ij}^{(ini)} = \begin{cases} 0 & \text{if } (i,j) \notin A_c, \\ \frac{1}{g_s(i)} & \text{if } (i,j) \in A_c. \end{cases}$$

donde $g_s(i)$ es el grado del nodo i en el subgrafo $G_s(V, A_c)$.

3.2. Distribute_Ants

Recibe como entrada el subgrafo $G_s(V, A_c)$, dos nodos terminales $i, j \in T$ y el valor r_{ij} que indica el nivel de conexión que deben tener i y j en la solución. En función de los grados de estos nodos en el subgrafo $G_s(V, A_c)$ y de r_{ij}, se determina la cantidad de hormigas a ubicar en los terminales i y j respectivamente. La forma de calcular estas cantidades viene dada por: $ants\_i = \left\lfloor \frac{g_s(i)}{r_{ij}} \right\rfloor \times$ ANT_FACTOR y $ants\_j = \left\lfloor \frac{g_s(j)}{r_{ij}} \right\rfloor \times$ ANT_FACTOR con $g_s(i)$ y $g_s(j)$ los grados de i y j en $G_s(V, A_c)$, y ANT_FACTOR es una constante elegida de antemano. La cantidad total de hormigas viene dada por $tot\_ants = ants\_i + ants\_j$.

3.3. Initialize_Trails

Recibe como entrada el subgrafo $G_s(V, A_c)$, y retorna la asignación inicial de rastros de feromona en dicho subgrafo. Inicialmente estos rastros son muy "tenues", incrementándose luego con el desplazamiento de las hormigas sobre dicho subgrafo. Sea W la matriz donde se almacena la intensidad de rastros de feromona sobre el subgrafo $G_s(V, A_c)$ y ΔW la matriz delta rastro donde se almacena el incremento de feromona sobre el subgrafo $G_s(V, A_c)$ luego de una unidad de tiempo. El procedimiento Initialize_Trails inicializa estas matrices de la siguiente forma:

$$W_{ij} = \begin{cases} \text{MIN\_TRAIL} & \text{if } (i,j) \in A_c, \\ 0 & \text{otherwise,} \end{cases} \quad \Delta W_{ij} = 0, \forall (i,j) \in A_c.$$

Procedure Search_Path(V,A_c,R,C,α,β,ρ,Q,i,j)

$num\_paths \leftarrow 0$; /* Inicializo la cantidad de caminos encontrados entre i y j */
$num\_iter \leftarrow 0$; /* Inicializo el número de iteraciones */
$best\_cost\_path \leftarrow \infty$; /* Inicializo costo de mejor camino encontrado */
$MP \leftarrow \text{Initialize\_Prob\_Trans}(V, A_c)$; /* Inicializo las probabiliades de transición */
$[W, \triangle W] \leftarrow \text{Initialize\_Trails}(V, A_c)$; /* Inicializo los rastros y los delta rastros */
$[ants\_i, ants\_j, tot\_ants] \leftarrow \text{Distribute\_Ants}(V, A_c, i, j, r_{ij})$; /* Distribución inicial de las hormigas */
while $(num\_iter \leq \text{MAX\_ATTEMPTS})$ **and** $(num\_paths < \text{MAX\_PATHS})$ **do**
  $ants\_location[1 \ldots ants\_i] \leftarrow i$; /* En el nodo i se colocan $ants\_i$ hormigas */
  $ants\_location[ants\_i + 1 \ldots ants\_j] \leftarrow j$; /* En el nodo j se colocan $ants\_j$ hormigas */
  $L \leftarrow \text{Initialize\_Tabu\_List}(i, j, ants\_i, ants\_j)$; /* Inicializo la estructura Tabu List */
  $\Delta W_{ij} \leftarrow 0, \forall (i,j) \in A_c$; /* Inicializo los delta rastros */
  for $t = 1$ **to** n **do** /* Para todo nodo */
    for $k = 1$ **to** tot_ants **do** /* Para toda hormiga en el tiempo t */
      $u \leftarrow ants\_location[k]$; /* Ubicación de la k-ésima hormiga */
      $v \leftarrow \text{Next\_Node}(V, A_c, MP, u)$; /* Movimiento de la k-ésima hormiga */
      $aux\_location[k] \leftarrow v$;
      $L_k \leftarrow \text{Insert\_Node}(L_k, v)$; /* Inserto en L_k el nodo elegido por la k-ésima hormiga */
      $ant\_trail \leftarrow \begin{cases} \frac{Q}{c_{uv}} & \text{Si se utiliza la clase Ant-Quantity,} \\ Q & \text{Si se utiliza la clase Ant-Density.} \end{cases}$; /* Computo el rastro de feromona dejado en la arista (u, v) por la k-ésima hormiga */
      $\Delta W_{uv} \leftarrow \Delta W_{uv} + ant\_trail$; /* Actualizo el delta rastro en (u, v) */
    end_for;
    $ants\_location \leftarrow aux\_location$; /* Actualizo la posición de las hormigas */
    $W_{ij} = (\rho \cdot W_{ij}) + \Delta W_{ij}, \forall (i,j) \in A_c$; /* Actualizo las intensidades de rastro */
    $MP \leftarrow \text{Update\_Prob\_Trans}(V, A_c, W, C, \alpha, \beta)$; /* Actualizo probabilidades de transición */
  end_for;
  $[p, cost\_path, OK] \leftarrow \text{Select\_Path}(L, i, j, T, C)$; /* Selecciono un camino entre i y j en L */
  if OK **then**
    if $(cost\_path < best\_cost\_path)$ **then** /* Actualizo el mejor camino */
      $num\_paths \leftarrow num\_paths + 1$;
      $num\_iter \leftarrow 0$;
      $best\_cost\_path \leftarrow cost\_path$;
      $p_{best} \leftarrow p$;
    else $num\_iter \leftarrow num\_iter + 1$;
    end_if;
  else $num\_iter \leftarrow num\_iter + 1$;
  end_if;
end_while;
return p_{best},num_paths;
end Search_Path;

Figura 2: Pseudocódigo de Search_Path.

3.4. Initialize_Tabu_List

Recibe como entrada los nodos $i, j \in T$ y las cantidades $ants\_i$ y $ants\_j$ de hormigas ubicadas en los nodos i y j respectivamente. Se inicializa la estructura tabu L donde se almacena el recorrido hecho por cada una de las hormigas sobre el subgrafo $G_s(V, A_c)$ en cada unidad de tiempo. La estructura se iniciliza mediante:

- $L_k \leftarrow \text{Insert\_Node}(L_k, i),\ \forall k \in 1 \ldots ants\_i$. El nodo i es el nodo inicial del camino a recorrer por la hormiga k.
- $L_k \leftarrow \text{Insert\_Node}(L_k, j),\ \forall k \in (ants\_i + 1) \ldots tot\_ants$. El nodo j es nodo inicial del camino a recorrer por la hormiga k.

3.5. Next_Node

Recibe como entrada el subgrafo $G_s(V, A_c)$, un nodo $u \in V$ sobre el cual hay una hormiga, y la matriz estocástica de transición MP. En base a la distribución en probabilidad dada por la fila $MP_{(u,\cdot)}$ se realiza un sorteo para determinar hacia que nodo adyacente al nodo u en $G_s(V, A_c)$ se moverá la hormiga.

3.6. Insert_Node

Recibe como entrada una lista L_k de la tabu list (donde se almacena el camino recorrido hasta el momento por la k-ésima hormiga) y un nodo $v \in V$ a ser agregado a la lista L_k. Se retorna la lista actualizada insertando el nodo u al final.

3.7. Update_Prob_Trans

Recibe como entrada el subgrafo $G_s(V, A_c)$, la matriz de rastros W, la matriz de costos C, los parámetros Ant System α y β elegidos, y la matriz estocástica de transición MP. En base a estas entradas, se actualiza la matriz MP de la siguiente forma:

$$MP_{ij} = \begin{cases} \frac{(W_{ij})^{\alpha} \times (\eta_{ij})^{\beta}}{\sum_{(i,k) \in A_c} \left((W_{ik})^{\alpha} \times (\eta_{ik})^{\beta}\right)} & \text{if } (i,j) \in A_c, \\ 0 & \text{otherwise.} \end{cases}$$

donde $\eta_{ij} = \frac{1}{c_{ij}}$ se le denomina visibilidad de la arista $(i, j) \in A_c$.

3.8. Select_Path

Recibe como entradas la estructura tabu L, dos nodos terminales $i, j \in T$, el conjunto de nodos terminales T, y la matriz C de costos de las aristas. Devuelve el camino $p \in L$ que satisface el Criterio de Selección de Camino especificado en el párrafo siguiente, el costo del camino seleccionado y además una variable indicatriz del éxito de la selección.

Sea $W^{i,j} = \{w^{i,j}\}$ el conjunto de caminos que unen los nodos terminales $i, j \in T$ determinado por un conjunto de hormigas en una instancia de ejecución del algoritmo Search_Path. El camino seleccionado, es aquel $\bar{w} \in W^{i,j}$ que satisface:

$$\frac{\text{COST}(\bar{w})}{\text{NTERM}(\bar{w}) + 2} = \text{mín} \left\{ \frac{\text{COST}(w^{i,j})}{\text{NTERM}(w^{i,j}) + 2}, w^{i,j} \in W^{i,j} \right\},$$

donde COST(w) es el costo del camino w y NTERM(w) es el número de terminales distintos de i y j presentes en w y para los cuales todavía existen requerimientos de conexión no satisfechos en la solución actual $G_s(N, A)$. La idea es seleccionar el camino que une los nodos terminales i y j que haya recorrido la mayor cantidad de nodos terminales con el menor costo posible. Los nodos terminales tomados en cuenta por NTERM($\cdot$) son aquellos para los cuales su requerimiento de conexión con algún otro nodo terminal todavía no ha sido satisfecho por la solución actual $G_s(N, A)$ (en fase de construcción). Es fácil ver que según este criterio, dados dos caminos de $W^{i,j}$, a igual costo se prefiere el que haya recorrido el mayor número de terminales, e inversamente, a igual número de terminales recorridos, se prefiere el que tenga menor costo.

4. Implementación y Tests de Performance

La implementación del algoritmo A_GSP se realizó en el lenguage de programación ANSI C. Las pruebas fueron realizadas en un equipo Sun SPARCStation 5 con 256 MB de memoria RAM y sistema operativo Solaris 7. Los rangos para los valores de los parametros α, β, ρ y Q fueron elegidos tomando como referencia la aplicación de Ant System a los problemas de ruteo de vehículos TSP, VRP, y MVRP [3, 4, 6, 12].

La fase experimental se realizó en dos etapas. En una primera etapa (Fase de Validación) el objetivo fue medir la sensibilidad de los parámetros α, β, ρ y Q, y seleccionar las combinaciones de valores para los cuales se obtuvieran los mejores resultados luego de aplicar el A_GSP a un conjunto de instancias del GSP de pequeñas dimensiones (menos de 20 nodos). Las instancias de validación fueron seleccionadas con la finalidad de testear topologías particulares que ponen a prueba la capacidad de construcción de una "buena"solución factible por parte del A_GSP. En todos los casos el A_GSP tuvo un muy buen desempeño, obteniéndose la solución óptima en todas las pruebas de validación. Las instancias de validación y los resultados experimentales obtenidos en dicha fase pueden encontrarse en [8].

Una vez concluida la Fase de Validación, seleccionamos las mejores combinaciones de valores de α, β, ρ y Q para utilizarlas posteriormente en instancias del GSP con topologías de mayores dimensiones (Fase de Análisis de Performance). Todas las instancias test utilizadas pueden encontrarse en [8]. Seguidamente presentamos algunas de las intancias del GSP que elegimos para testear la performance del A_GSP.

La selección se realizó buscando modelos reales de optimización de redes telefónicas y de comunicación, donde los grafos asociados tengan un número considerable de nodos y aristas. Esto permite estudiar el desempeño del algoritmo en problemas con redes de mediana y gran escala. No hay publicadas muchas aplicaciones reales que requieran niveles de conexión mayores a 2 entre pares de nodos terminales. En el diseño de redes de fibras ópticas, en general, alcanza con exigir que la red sea 2-arista-conexa para lograr un nivel de confiabilidad elevado. Algunas de las (pocas) aplicaciones reales donde se utiliza el modelo GSP con requerimientos de conexión mayores que 2 entre nodos terminales son redes con fines militares [7]. Los siguientes son tres de los casos de pruebas experimentales utilizados en la Fase de Análisis de Performance.

(1) El diseño de una red 2-arista-conexa basada en una topología de grilla doble con 33 nodos terminales, 87 nodos de Steiner y 268 aristas. Dicha instancia esta referenciada en [8, 10]. Se conoce una solución óptima computada mediante la aplicación de una algoritmo Backtracking paralelizado [10].

(2) El HSODTN (High Speed Optical Data Transmission Network), red de comunicación de [illegible] tecnología, que conecta diferentes sitios de un portaaviones permitiendo

la comunicación entre diferentes puntos estratégicos. La razón por la cual se desea un alto grado de confiabilidad en la red es obvia: ésta debe ser capaz de continuar funcionando al producirse daños en un enfrentamiento en batalla. Presentamos aquí una topología de red (versión reducida) donde se modela las posibles líneas de comunicación de fibra óptica entre diferentes sitios del buque, y donde los puntos estratégicos se modelan como nodos terminales. Se desea encontrar un subgrafo 3-arista-conexo de costo mínimo que cubra el conjunto de nodos terminales. Esta instancia y otras similares de mayor escala pueden encontrarse en [7, 9, 11].

(3) Una instancia genérica del GSP, donde el grafo asociado tiene 48 nodos del tipo Steiner, 13 nodos terminales y diferentes requerimientos de conexión entre pares de nodos terminales en el rango 1..3. Dicha instancia, así como su solución óptima, puede encontrarse en [8]. La matriz de requerimientos R viene dada por:

$$R = \begin{pmatrix} 0 & 2 & 1 & 3 & 2 & 2 & 3 & 2 & 1 & 2 & 2 & 2 & 2 \\ 2 & 0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 2 & 2 & 2 \\ 1 & 1 & 0 & 2 & 2 & 2 & 2 & 3 & 3 & 1 & 1 & 2 & 2 \\ 3 & 1 & 2 & 0 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 \\ 2 & 2 & 2 & 2 & 0 & 1 & 1 & 1 & 2 & 2 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 & 0 & 2 & 2 & 3 & 3 & 2 & 1 & 2 \\ 3 & 2 & 2 & 2 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & 2 & 1 & 2 & 1 & 0 & 3 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 2 & 2 & 3 & 2 & 3 & 0 & 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 1 & 2 & 3 & 2 & 2 & 1 & 0 & 2 & 2 & 2 \\ 2 & 2 & 1 & 2 & 1 & 2 & 2 & 2 & 1 & 2 & 0 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 2 & 3 & 0 & 2 \\ 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & 2 & 3 & 2 & 0 \end{pmatrix}.$$

La Figura 3 muestra sus topologías originales y sus respectivas soluciones óptimas. Los nodos blancos representan los nodos terminales y los nodos negros los nodos de Steiner.

5. Resultados Numéricos y Conclusiones

Los parámetros utilizados en las pruebas del A_GSP fueron:

- MAX_ITER es el número máximo de intentos realizados por el A_GSP para encontrar una solución factible al problema. Fue seteado con valor 3 en todas las corridas de performance.
- MAX_SOL es el número máximo de soluciones factibles buscadas por el A_GSP para encontrar una solución de bajo costo al problema. Fue seteado con valor 10 para estas instancias.
- MAX_ATTEMPTS es el número máximo de intentos realizados por el Search_Path para encontrar un camino entre un par de nodos terminales. Fue seteado con valor 2 en todas las corridas de performance.
- MAX_PATHS es el número máximo de caminos computados por Search_Path entre un par de nodos terminales. Fue seteado con valor 3 en todas las corridas de performance.

Para el procedimiento Search_Path, se probón con distintos valores para los parámetros, elegidos en base a la literatura sobre Ant Systems:

- $\alpha \in \{,5, 1, 2, 5, 10\}$ con α la sensibilidad del rastro,
- $\beta \in \{,5, 1, 2, 5, 10\}$ con β la sensibilidad de distancia,
- $\rho \in \{,5, ,7, ,9\}$ con ρ el coeficiente de evaporación,
- $Q \in \{10, 100, 10000\}$ con Q la cantidad de feromona dejada por una hormiga por unidad de tiempo,
- ANT_FACTOR=4, es el parámetro fijo utilizado por Distribute_Ants en el cálculo de la cantidad de hormigas a ubicar en cada nodo terminal.

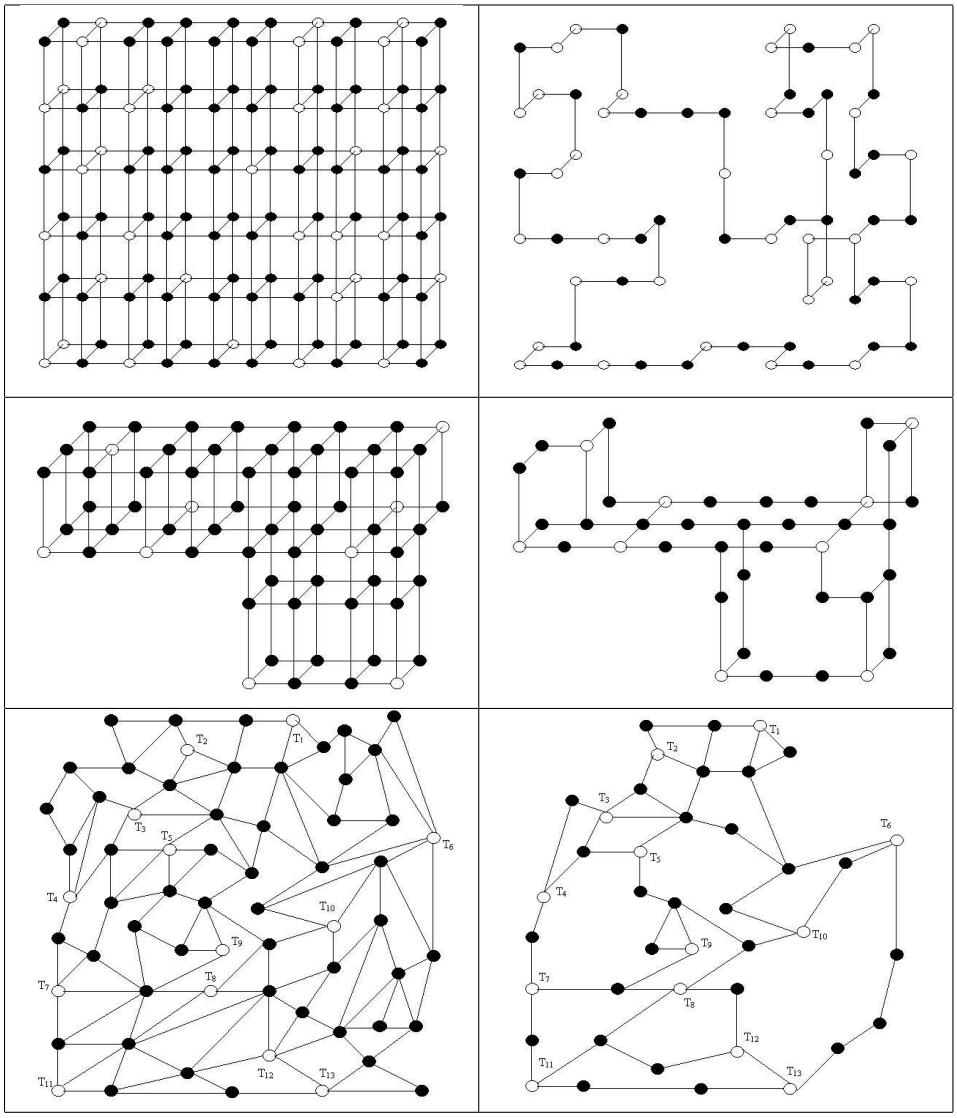

Figura 3: Instancias 1, 2 y 3 y sus soluciones óptimas.

La Tabla 1 muestra para cada una de las instancias anteriores, algunos de los resultados obtenidos luego de realizar las corridas con diferentes combinaciones de los parámetros. Las columnas de esta tabla son las siguientes:

- NP es el numero de instancia,
- NA es el número máximo de hormigas usado para la búsqueda de un camino entre un par de nodos terminales,
- NF es el número de soluciones factibles encontradas en la corrida,
- CS es el costo de la mejor solución factible encontrada,
- CO es el costo de la solución óptima global,
- GAP es el porcentage relativo de error y se define como: $\text{GAP} = 100 \times \left(\frac{\text{CS}-\text{CO}}{\text{CO}}\right)$.

NP	α	β	ρ	Q	ANT_FACTOR	NA	NF	CS	CO	GAP
(1)	0.5	0.5	0.7	10	4	8	2	151	145	4.13
(1)	0.5	1	0.7	10	4	8	3	149	145	2.75
(1)	0.5	2	0.7	10	4	8	2	149	145	2.75
(1)	1	5	0.7	100	4	8	4	145	145	0
(1)	2	10	0.7	100	4	8	4	145	145	0
(1)	2	10	0.9	100	4	8	4	145	145	0
(2)	0.5	0.5	0.7	10	4	8	2	80	74	8.10
(2)	0.5	1	0.7	10	4	8	2	79	74	6.75
(2)	1	2	0.7	100	4	8	2	77	74	4.05
(2)	1	5	0.7	100	4	8	2	77	74	4.05
(2)	1	10	0.7	100	4	8	3	74	74	0
(2)	2	5	0.7	100	4	8	2	74	74	0
(2)	2	10	0.7	100	4	8	2	74	74	0
(2)	2	10	0.9	100	4	8	2	74	74	0
(3)	1	2	0.7	10	4	16	4	109	98	11.22
(3)	2	2	0.7	100	4	16	3	105	98	7.14
(3)	2	5	0.7	100	4	16	4	103	98	5.10
(3)	2	10	0.7	100	4	16	5	101	98	3.06

Cuadro 1: Tabla con resultados de A_GSP

Analizando los resultados obtenidos en las diferentes corridas de performance asociadas a cada una de las instancias test, se observó lo siguiente:

a) Los parámetros ρ y Q tienen influencia directa en la cantidad total de rastro dejado por las hormigas en cada iteración de movimiento sobre el grafo.

b) Los parámetros α, β y Q afectan el tiempo requerido para que las hormigas elijan el mismo camino entre un par de nodos terminales (comportamiento uniforme).

c) La búsqueda de "buenos caminos"(de acuerdo al Criterio de Selección de Camino) entre un par de nodos terminales es muy sensible a la elección de β (sensibilidad de distancia). Ligado a esto, el parámetro β también es crítico para la obtención de buenas soluciones factibles por parte del A_GSP.

d) En las corridas en las cuales se alcanzó el óptimo con un coeficiente de evaporación $\rho = 0{,}7$, también se alcanzó con $\rho = 0{,}9$; ambos parecen ser buenos valores para este parámetro.

e) Los mejores resultados se obtuvieron con el juego de parámetros: $\alpha = 2$, $\beta = 10$, $\rho = 0{,}7$ y $Q = 100$, alcanzándose la solución óptima en todos los problemas excepto el problema test 3 presentado aquí. En el problema 3 en el mejor caso se obtuvo una solución factible con una diferencia porcentual de costo de un 3,06 % con respecto al costo de la solución óptima.

f) La eficiencia computacional del A_GSP esta fuertemente ligada a las estrategias utilizadas por el procedimiento Update_Matrix para actualizar la matriz M donde se almacena los requerimientos de conexión sin sastifacer en la solución actual $G_s(N, A)$.

Es de destacar el esfuerzo realizado en la Fase de Validación para ajustar adecuadamente los parámetros α, β, ρ y Q con la finalidad de poder obtener "buenas"soluciones factibles en la Fase de Analisis de Performance. Una vez realizado el refinamiento de estos parámetros, las corridas realizadas en la Fase de Analisis de Performance mostraron ser exitosas, obteniéndose la solución óptima en casi todos los problemas test salvo en la instancia 3 presentada aquí, en la cual se alcanzó una solución subóptima con porcentaje relativo de error inferior a 3,07 %.

Referencias

[1] Ajit Agrawal, Philip Klein, and R. Ravi, "When trees collide: An approximation algorithm for the Generalized Steiner Problem in Networksreport CS-90-32 (1994), Department of Computer Science, Brown University.

[2] M. Baïou, "Le problème du sous-graphe Steiner 2-aréte-connexe: approche polyédrale", Phd. Thesis, University of Rennes I (1996).

[3] A. Colorni, M. Dorigo and V. Maniezzo, "Distributed Optimization by Ant Colonies", ECAL91-European Conference on Artificial Life, Paris, France, pages 134-142.

[4] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini and M. Trubian, "Heuristics from Nature for Hard Combinatorial Optimization Problems", IFORS (1996), published by Elsevier Science Ltd.

[5] P. Crescenzi and V. Kann, "A compendium of NP optimization problems", http://www.nada.kth.se/~viggo/problemlist/compendium.html, Consultado julio 2003.

[6] M. Dorigo and Luca Maria Gambardella, "Ant-Q, A Reinforcement Learning Approach to Combinatorial Optimization", technical report 95-01, IRIDIA, Brussels Open University.

[7] M. Grötschel, C.L. Monma, and M. Stoer, "Polyhedral and computational investigations for designing communication networks with high survivability requirements", Operations Research 43 (1995).

[8] F. Robledo, Diseño Topológico de Redes : casos de estudio "The generalized Steiner problem.ªnd "The Steiner 2-Edge-Connected subgraph problem". Master Thesis. InCo, PEDECIBA Informática, (2000). 147 p. technical report RT 00-08, Facultad de Ingeniería, Universidad de la República, J. Herrera y Reissig 565, Montevideo, URUGUAY.

[9] F. Robledo and O. Viera, "An Ant-System algorithm for the Generalized Steiner Problem with edge-connectivity", Research Report PI 1503 (http://www.irisa.fr/bibli/publi/pi/1503/1503.html), IRISA, Rennes, France (2002).

[10] F. Robledo and O. Viera, "A parallel algorithm for the Steiner 2-edge-survivable network problem", Research Report PI 1504, (http://www.irisa.fr/bibli/publi/pi/1504/1504.html), IRISA, Rennes, France (2002).

[11] M. Stoer, "Design of Survivable Networks", Lecture Notes in Mathematics, ISBN 3-540-56271-0, ISBN 0-387-56271-0, Springer-Verlag, (1996).

[12] Eric. D. Taillard, "Ant Systems", technical report 05-99, IDSIA, Switzerland (1999).

[13] P. Winter, "Generalized Steiner problem in series-parallel networks", Journal of Algorithms 7 (1986), pages 549-566.

[14] P. Winter, "Steiner problem in networks: A survey", Networks 17 (1987), pages 129-167.